Mining Recurring Concepts in a Dynamic Feature Space

João Bártolo Gomes, Mohamed Medhat Gaber, Pedro A. C. Sousa, and Ernestina Menasalvas

Abstract— **Most data stream classification techniques assume that the underlying feature space is static. However, in real-world applications the set of features and their relevance to the target concept may change over time. In addition, when the underlying concepts reappear, reusing previously learnt models can enhance the learning process in terms of accuracy and processing time at the expense of manageable memory consumption. In this paper, we propose mining recurring concepts in a dynamic feature space (MReC-DFS), a data stream classification system to address the challenges of learning recurring concepts in a dynamic feature space while simultaneously reducing the memory cost associated with storing past models. MReC-DFS is able to detect and adapt to concept changes using the performance of the learning process and contextual information. To handle recurring concepts, stored models are combined in a dynamically weighted ensemble. Incremental feature selection is performed to reduce the combined feature space. This contribution allows MReC-DFS to store only the features most relevant to the learnt concepts, which in turn increases the memory efficiency of the technique. In addition, an incremental feature selection method is proposed that dynamically determines the threshold between relevant and irrelevant features. Experimental results demonstrating the high accuracy of MReC-DFS compared with state-of-the-art techniques on a variety of real datasets are presented. The results also show the superior memory efficiency of MReC-DFS.**

I. Introduction

THE availability of data streams in information systems opens an opportunity to apply new data stream mining techniques that will support intelligent decision making in a wide range of applications. This paper is framed in the scope of data stream classification, which aims to learn a classification model from a stream of training records and apply such model to predict the class of unlabeled records with high accuracy.

Many existing data stream classification techniques address the problem of changing concepts, yet other important challenges have received far less attention, for instance, recurring changes in concept, integration of context information and feature space evolution.

Over the past years, researchers have shown an increasing interest in the problem of concept drift [7], [14], [24], [25], [29], [32], and propose that data stream classification algorithms must recognize and adapt to concept changes by continuously learning the different time-changing concepts [7], [25], [32]. Many algorithms have been proposed in response to this interest [4], [5], [7], [12], [14], [17], [20], [24], [25], [29]. However, some other related challenges have received far less attention, for instance, most of these techniques assume that the underlying feature space is static, yet in real-world applications the set of features and their relevance to the target concept may change [15], [28]. In addition, when these concepts reappear, reusing previously learnt models can enhance the learning process in terms of accuracy and processing time [1], [6], [16], [22], [31].

As an illustrative example, imagine that it is required to learn from a stream of Twitter posts (tweets) which ones are interesting to the user. However, the user's interest may change over time due to changes in preference or context. To the target function that the classification algorithm learns from training data (e.g., tweets labeled as interesting), we will refer as the target/underlying concept. The context is considered to be the information about the situation when the data was observed/recorded but that is not a direct predictor of the class label (the formal definition of context as used in this paper is given later).

The relevant feature space of the records in the data stream may change over time [15], [28]. For example, in a stream of tweets where each word is represented as a feature, it is impossible to know in advance which words will appear over time, which ones are more relevant to the target concept, and consequently what the best feature space to represent the stream is. Using a very large vocabulary of words is clearly inefficient, as most of the words will likely be redundant because only a small subset of words is finally useful for classification. Over time, it is also likely that new important features appear and that previously relevant features become less important, which in turn leads to change in the subset of relevant features. Such change in the feature space is related to the problem of concept drift, as the target concept may change due to changes in the relevance of the available features. However, most existing approaches are not able to learn in a dynamic feature space. Katakis *et al.* [15] proposed the usage of an incremental feature selection to asses feature predictiveness over time, and to use a feature-based classifier that can process records in such dynamic feature space.

Mostly concepts are recurrent; this means that a previously seen concept may reappear in the future [6], [16], [29], [31] and likely in a similar context [11], [29]. This problem is a particular type of concept drift that is common in many real-world domains [11], [29]. However, only a few approaches explored it [6], [16], [29], [31]. For instance, weather prediction models change according to the seasons, and in recommender applications user consumption patterns change over time due to fashion, economy, spatial-temporal situation and/or any other context [11], [25], [29]. In these applications, concept changes are generally due to context changes. If context information is available, it can be used to better understand recurring concept changes. However, only a small number of techniques explored context information to deal with recurring concept changes [1], [9], [11], [29].

In this paper, we address this research problem by proposing mining recurring concepts in a dynamic feature space (MReC-DFS), a data stream learning system to address the issues that arise from learning in a dynamic feature space while simultaneously exploiting incremental feature selection to reduce the size of the models, and thus minimize the memory consumption of the learning system.

MReC-DFS is able to detect and adapt to concept changes using the performance of the learning process and contextual information. To deal with recurring concepts, stored models are combined in a dynamic weighted ensemble. The weighting is computed based on model performance and associated context information. The models are learnt using an incremental learning algorithm that can process records with a dynamic feature space. Incremental feature selection is performed to reduce the combined feature space. Instead of storing a model that represents the full static feature space, only the most relevant features to the corresponding target concept are kept. This reduces the memory consumption of the approach and makes it possible to work with data streams in which previously would not been feasible.

This paper presents several contributions. First, we propose MReC-DFS, a data stream learning system which addresses the following issues: 1) learning and classifying records (possibly infinite) from a data stream with a dynamic feature space; 2) concept drift and recurrence; and 3) integration of context information. To the best of our knowledge, this is the first work to deal with recurring concepts in a dynamic feature space. Second, we propose a feature selection method that adaptively defines the threshold above which features are considered relevant. The method uses the desired percentile of the scores given by the feature evaluation method as a threshold. Finally, we evaluate MReC-DFS using several real-world data streams. The experimental results demonstrate the feasibility of MReC-DFS, and its efficiency in terms of accuracy, memory consumption and number of processed records.

The rest of this paper is organized as follows. In Section II, we summarize related work on concept drift, context-aware approaches in recurring concepts and learning in a dynamic feature space, which is followed in Section III by the preliminaries of the approach where the motivation, challenges and problem definition are stated. In addition, in Section IV, we propose MReC-DFS, with detailed description of its components. Section V introduces the experimental setup and the datasets used to evaluate MReC-DFS. This is followed by a detailed discussion of the results of our evaluation. Finally, in Section VI, our conclusion and possible topics for future work are presented.

II. Related Work

The proposed approach deals with learning recurring concepts from a data stream in a dynamic feature space. Previously learnt models are associated with context and combined into an ensemble that classifies incoming unlabeled records when the underlying concept is recurrent. Consequently, we review firstly methods that address the problems of: recurring concepts; context and context-aware approaches. Then we review the few existing methods to deal with a dynamic feature space. Finally, the contribution of our proposal in relation to existing techniques is discussed.

A general review of the literature related to the problem of concept drift can be found in [25] or more recently in [32].

A. Recurring Concepts

Most of the approaches in the literature to deal with concept drift [25], [32] have not addressed the problem of recurring concepts, and they have to relearn them as if the concepts are new, and not recurring.

Ensemble approaches, include the DWM algorithm [17] that dynamically builds and deletes weighted classifiers in response to changes in performance. The models are created at different time steps so they use different training set of records. The final prediction is obtained as a weighted vote of all the classifiers. The weights of all the models that misclassified the record are decreased by a multiplicative constant β. If the overall prediction is incorrect, a new expert is added to the ensemble with a weight equal to the total weight of the ensemble. Bifet *et al.* [4] have proposed two bagging methods for evolving data streams: the ASHT Bagging using trees of different sizes, and ADWIN Bagging using a change detector to decide when to discard underperforming experts from the ensemble. Ramamurthy and Bhatnagar [22] presented an ensemble approach that exploits concept recurrence, they use a global set of classifiers learnt from sequential data chunks. If no classifier in the ensemble performs better than the error threshold, a new classifier to represent the current concept is learnt and stored. The classifiers with better performance on the most recent data form part of the ensemble for labeling new records. Similarly, in [16] an ensemble is used but incremental clustering is performed to maintain information about historical concepts, the proposed framework captures batches of examples from the stream into conceptual vectors. Conceptual vectors are clustered incrementally by their distance and for each cluster a new classifier is learnt. Classifiers in the ensemble are learnt using the clusters. Recently [5] proposed Learn++.NSE, an extension of [20] for nonstationary environments, Learn++.NSE is also an ensemble approach that learns from consecutive batches of data without making any assumptions on the nature or rate of drift. The classifiers are combined using dynamic weight majority and the major

novelty is on the weighting function that uses the classifiers time-adjusted accuracy on current and past environments. To deal with resource constraints [13] proposes a novel algorithm to manage a pool of classifiers when learning recurring concepts.

Sophisticated approaches that use drift detection [7] have also been proposed to address concept recurrence, such as [6], [31]. These approaches store learnt models and reuse them when a similar concept reappears in the stream, thus avoiding the effort to relearn a previously observed concept. The method proposed by Yang *et al.* [31] involves using a proactive approach to recurring concepts by reusing a concept from the history of concepts. This history of concepts is represented as a Markov chain and allows selecting the most probable concept according to a given transition matrix. The approach proposed by Gama and Kosina [6] uses the drift detection method presented in [7] to identify stable concepts and memorizes learnt classifiers that represent these concepts. After change is detected in situations of recurrence, referees are used to choose the most appropriate classifier to reuse (i.e., the referee prediction about the applicability of the classifier is greater than a predefined threshold). Exponentially weighted moving average for concept drift detection [23] is a recent work on drift detection which uses a chart to monitor the misclassification rate of the data stream classifier. RCD [10] is a recent recurring concept drift framework that uses a nonparametric multivariate statistical tests to check for recurrence. [18] proposes a semisupervised recurring concept learning algorithm that takes advantage of unlabeled data.

However, neither of these approaches explores the usage of context information, nor can it learn in a dynamic feature space.

B. Context-Aware Approaches

Context dependence has been recognized as a problem in several real-world domains [11], [26], [29]. Turney [26] was among the first ones to introduce the problem of context in machine learning, when he presented a formal definition in which the notions of primary, contextual and context-sensitive features were introduced. Such notions are based on a probability distribution for the observed classes given the features.

Widmer [29] exploited what is referred as contextual clues (based on the Turney [26] definition of primary/contextual features) and proposes a meta-learning method to identify such clues. Contextual clues are context-defining attributes or combinations of attributes which values are the characteristics of the underlying concept. When more or less systematic changes in their values are observed this might indicate a change in the target concept. The method automatically detects contextual clues on-line, and when a potential context change is signaled, the knowledge about the recognized context clues is used to adapt the learning process in some appropriate way. However, if the hidden context is not represented in the contextual clues, this is, if the reason behind the change is not represented in the feature space, it is not possible to detect and adapt to the change.

The approach of conceptual clustering proposed by Harries [11], identifies stable hidden contexts from a training set by clustering the instances assuming that similarity of context is reflected by the degree to which instances are well classified by the same concept. A set of models is constructed based on the identified clusters. This approach has been proved to work well with recurring concepts and real-world problems. However, its main drawback is the off-line training required to obtain the conceptual clusters, as these could lead to inaccuracy with concepts or patterns that were not seen during training.

In the approach we proposed in [1] and [9], context integration shares the motivation with the approach presented in [11] where the method infers periods when the context is stable (from available context features), that are described as contextual clusters. However, [1] proposed an on-line method that learns context-concept relations from the concept history. In addition, the proposed method does not require the partition of the dataset into small batches as the concept representations are learnt from an arbitrary number of records, as determined by the drift detection method. To improve [1] and [9], which rely on a single classifier to deal with recurring concepts, the usage of ensembles has been proposed to deal with these in [2]. Nevertheless, [1], [2], and [9] only consider records from a static feature space and all the features are saved, which limits the applicability of the approach to dynamic feature space data or high dimensional feature space data.

C. Dynamic Feature Selection

Katakis *et al.* [15] was among the first to introduce the problem of a dynamic feature space over time in data streams. They reviewed the different approaches for feature selection (i.e., filter or wrapper approaches), and for the purpose of online feature selection considered the filter approach to be more adequate for online processing due to its lower computational costs. Such filters evaluate the predictive power of each feature, which allows discriminating and selecting the N most predictive ones. The filters are based on cumulative statistics (i.e., contingency tables) of the number of times a feature appears in each distinct class. Updating such statistics is incremental by nature, which makes the method suited for data stream processing.

Consequently, a minimum degree of relevance for the features can be established by simply selecting the N most relevant features. In our framework, this allows making the distinction between predictive and irrelevant features. The predictive score of each feature can be computed using popular methods, such as, the information gain, χ^2 or mutual information [15], [28]. Katakis *et al.* [15] proposed a technique that involves a feature ranking method which allows the selection of the relevant features. In addition, they proposed the usage of an incremental feature-based learning algorithm as it can deal with the distinct feature space over time (i.e., dynamic feature space). Wenerstrom and Giraud-Carrier [28] proposed a technique, feature adaptive ensemble (FAE), which also applies incremental feature ranking and selection, but an ensemble of classifiers is used to classify unlabeled records.

Their experimental results show that FAE achieves relatively better performance than the Katakis *et al.* [15] approach. In a recent work, Masud *et al.* [19] proposed a data stream classification technique (DXMiner) to address the problem of novel classes. DXMiner uses the deviation weight to deal with the feature ranking and selection problem.

In addition, in FAE, when the feature space of an unlabeled record is different from the feature space of the classification model, the unlabeled record uses only the features available in the classification model, which is a lossy conversion. In contrast, DXMiner uses lossless conversion that exploits all the available features which is useful to detect novel classes.

Similarly, our system deals with a dynamic feature space. In the proposed approach the features are not just considered relevant/nonrelevant, but divided into predictive, irrelevant and contextual as defined in Section III-C.2. To the best of our knowledge, the approach proposed in this paper is the first to deal with recurring concepts in a dynamic feature space. In addition, we propose a feature selection method that adaptively selects (for each concept), the threshold that separates the predictive/relevant features from the irrelevant ones (refer to Section IV-B.1 for details).

III. Preliminaries

This section provides the necessary background for our MReC-DFS system. We start by motivating finding a solution to the problem, followed by enumeration of the challenges that face finding such a solution. Finally, we formally define the problem.

A. Motivation

Data stream mining algorithms can only keep in memory a bounded number of records, as a result of the massive data volumes (possible infinite) that characterize real-world data streams. Therefore, each training record should be processed only once, as opposed to traditional data mining algorithms, where multiple passes over the data are common. Consequently, in stream mining approaches it is usual that the classification model is learnt incrementally. In addition, after the first training records are processed it is possible to predict the class of unlabeled records. An anytime classification scenario is assumed and the accuracy of classification model is expected to increase as the number of processed training records grows. However, this will not happen if the underlying data distribution changes, as the classification accuracy will decrease. Therefore, to continuously maintain the high quality of the results, it is important that the data stream classification algorithm not only learns incrementally but also detects and adapts to the changes in the underlying concept [25].

The changes in the underlying concept can be the result of:

1) context changes, either hidden and explicit [7], [11], [25], [29];
2) changes in the underlying feature space [15], [28];
3) recurring concepts, a particular type of concept change [6], [16], [29], [31].

The type of change is also a property of change and it can be sudden or gradual [32].

The causes of change are somehow related, as usually recurring concepts reappear associated with the same context [11], [29]. In such cases recognizing an already learnt concept might improve the adaptation to change by avoiding relearning that concept from scratch [1], [6], [16], [29], [31]. This approach has been explored in [1] and [9], where the previously seen concepts are saved with the associated context. However, changes in the underlying feature space and their relevance to the target concept were not considered.

B. Challenges

In this paper, we propose the MReC-DFS learning system to learn time changing concepts from a data stream in which the feature space can be dynamic. In addition, MReC-DFS exploits the fact that the features relevance can be different for each target concept. This way when storing previously learnt models, only the most relevant features for the given concept are saved. For example, if the target concept is the function that classifies tweets as interesting for a user interested in sports, the most relevant words for this concept will be different than when the user is interested in classical music.

Consequently, saving only the most relevant features when dealing with recurring concepts results in a more compact representation of the learnt concepts, making the approach suitable for a wide range of real application scenarios while reducing its space complexity.

In this paper, we have identified and addressed major challenges when learning recurring concepts in a dynamic feature space. They are:

1) measure the relevance of the features in an online scenario;
2) select the most relevant features to represent concepts in a compact form without sacrificing good accuracy;
3) detecting and adapting to recurring concept changes, particularly when the concepts may have different feature spaces;
4) exploit context information associated with concepts and their feature space.

C. Problem Definition

In defining the problem, we will follow the general to specific approach, from providing the necessary definitions for learning a classifier from changing streaming data to providing those definitions required for the comprehension of leaning recurrent concepts.

1) Learning a Classifier From Changing Data Streams: Let D be the data stream of training records $X_i = (\vec{x}_i\ ,\ y_i)$ with $x_i \in X_t$ (feature space at time t) and $y_i \in Y$, that arrive sequentially, where $\vec{x}_i$ is a vector of attribute values (nominal or numeric) and y_i is the class label (discrete) for the ith record in the stream. These records are processed by a base learner to incrementally train a classification model m that can be used to predict the class label of a record $\vec{x} \in X_t$, such that $m(\vec{x}) = y \in Y$. The base learner should be able to handle records in a dynamic feature space X_t, which means that each feature should contribute independently in the resulting model m.

The overall goal of learning a classifier is that the trained model m minimizes the number of prediction errors.

A stable concept can be learnt when the records of a given period (or set) k (with an arbitrary number of records, with X_k as the combined feature set for period k) are independently identically distributed according to a distribution $P_k(x, y)$. In situations of concept change, $P_k(x, y) \neq P_{k+1}(x, y)$. Note that a change in the feature space from period k to $k + 1$ can cause a change in the distribution and thus a concept change. To minimize the number of prediction errors, m must be adapted to represent the new concept.

2) Dynamic Feature Space Definitions: Let X be the space of all features (nominal and numeric) and its possible values and Y the set of possible (discrete) class labels. Let X_t be the space of features that occur in the stream at time t and $X_t \subseteq X$. Please note that in this section and throughout this paper we used the term feature where in the literature it is used interchangeably with the term attribute.

According to general idea of Turney [26] and to Widmer [29], here for consistency and to introduce the notion of irrelevant features we define:

Definition 1: (Predictive feature-values). A feature-value combination $f_i : v_{ij}$ is predictive if $\mathrm{P}(c_k|f_i = v_{ij})$ is significantly different from $\mathrm{P}(c_k)$ for some class c_k.

Definition 2: (Predictive features). A feature f_i is predictive if one of its values v_{ij} (i.e., some feature-value $f_i : v_{ij}$) is predictive.

Definition 3: (Contextual feature-values). A feature $f_i : v_{ij}$ is contextual if it is predictive of predictive features-values, i.e., if $\mathrm{P}(f_k : v_{kl} \mid f_i = v_{ij})$ is significantly different from $\mathrm{P}(f_k : v_{kl})$ for some feature-value $ak{:}v_{kl}$ that is predictive.

Definition 4: (Contextual features). A feature f_i is contextual if one of its feature-values v_{ij} is contextual.

Definition 5: (Irrelevant attributes). A feature f_i is irrelevant if none of its values v_{ij} is not predictive or not contextual.

Such notions are based on a probability distribution for the observed classes given the features. However, when the probability distribution is unknown it is often possible to use background knowledge, as suggested by Turney [26], to distinguish between predictive and contextual features. In this paper, the same approach is followed, as the system processes is defined initially as contextual features in a meta-learning level and primary/irrelevant features are treated in the base learning level using feature evaluation and selection. In addition, the distinction between primary and irrelevant must be evaluated periodically as the underlying distributions may change over time. Therefore, one of the contributions of this paper is how to select the most predictive features for a certain concept. The process of feature selection proposed for MReC-DFS is described in detail in Section IV-B.

3) Recurring Concepts in a Dynamic Feature Space: A recurring concept change happens when the records from a period k are generated from the same distribution as a previously observed period $P_k(x, y) = P_{k-j}(x, y)$. In these particular concept changes, a model m_k learnt from a certain period k can be saved and then reused. This has been shown to improve the on-line learning process due to the fact that it is no longer required to learn from scratch a previously learnt concept. In addition, this approach reduces the number of training records that need to be processed, when we compare them with approaches that do not consider recurrence (Rec) (i.e., forget old models).

Finally, the model m_k instead of storing all the features X_k saves only the most relevant ones. This explores the advantages of incremental feature selection [15], such as reducing the space complexity and computational cost of the approach while improving the model generalization power and the quality of its results.

IV. MReC-DFS Data Stream Learning System

In situations of concept recurrence, anticipating to the reappearing concept can improve the learning process efficiency [1], [9]. Consequently, we propose to continuously store learnt models, representing the different underlying concepts in the data stream with dynamic feature space, and associate context information with these models. For each model only the most predictive features are kept, incremental feature evaluation and selection is used to determine each feature predictiveness for the current concept. The learnt models are combined in an ensemble that is used to classify incoming unlabeled records of a reappearing concept [2]. The weights of the ensemble and its member classifiers are calculated dynamically, considering their performance and associated context. When the models take part in the ensemble or are evaluated there is also the need to integrate their selected feature spaces with the current records feature space.

MReC-DFS is organized into a two-level framework:

1) base learner level where: a) an incremental algorithm learns the underlying concept, building a classification model and b) features are ranked periodically, the incremental algorithm used must be able to deal with a dynamic feature space. Thus, we will consider base learner, which assumes that each feature contributes independently to the classification. Note that this assumption may be a limitation as there are some learning problems (e.g., XOR) that require feature interaction to solve;
2) meta-learning level where: a) detection and adaptation to concept changes; b) the context-concept relations are learnt and used to deal with the recurring concepts; and c) classify unlabeled records using information learnt from different models and context.

Fig. 1 shows the learning process and its components. This continuous learning process consists of the following steps.

1) Process the incoming records from the data stream using an incremental learning algorithm (base learner) to obtain a decision model m capable of representing the underlying concept, and classify unlabeled records. The base learner must be able to deal with the dynamic feature space of the incoming records.
2) Context records are associated with the current model m. The history of context-concepts relations will be referred to as context-concepts relations history.

3) Periodically evaluate features into predictive/irrelevant.
4) A drift detection method that monitors the error-rate of the learning algorithm [7]. When error-rate goes above predefined levels the drift detection method signals a warning (possible drift) or drift.
5) When change is detected two situations are possible: 1) the underlying concept is new (i.e., no equivalent concept is represented in the classifiers repository) and the base learner will learn the new underlying concept by processing the current incoming labeled records. The incremental classifier that is being learnt will also classify the incoming unlabeled records as anytime classification is assumed. The evaluation and selection of features occurs here and 2) the underlying concept is recurrent (i.e., has been learnt previously). In this situation the classifiers in the repository that represent the underlying concept, are integrated in the ensemble that is created to classify incoming unlabeled records.

The base learner algorithm is described in Section IV-A. Dealing with the dynamic feature space is detailed in IV-B. Context representation and similarity are presented in Section IV-C. Concept representation, context-concepts relations history and the measure of similarity between models, used to check if different models represent the same concept, are discussed in IV-D. Also in the same section, subsections for model storage and ensemble weighting are presented. To detect when drift occurs a drift detection method is used [7]. This method is briefly summarized in Section IV-E. Finally, in Section IV-F the pseudo-code of the learning process is presented.

A. Base Learner

The base learner is used to learn a model that represents the data stream underlying concept. Any classification algorithm able to learn incrementally and that treats each feature independently can be used for this task. The base learner can be selected according to the nature of data to be mined, choosing the algorithm that best suits it (e.g., high accuracy, handles noise, memory consumed, and faster processing time). For our prototype we use the Naive Bayes (NB) algorithm as the base learner, because it represents the concepts in a compact form (i.e., results in memory efficiency), is incremental, can deal with a dynamic feature space (i.e., each feature is treated independently) and has shown good results as a base learner [6], [16], [29].

The NB classifier provides a simple, incremental and efficient approach to learn probabilistic knowledge. From the Bayes theorem, the probability that an record $\vec{x} \in X$ belongs in class $y \in Y$

$$P(Y = y|X = \vec{x}) = \frac{P(Y = y)P(X = \vec{x}|Y = y)}{P(X = \vec{x})}. \quad (1)$$

It is straightforward to estimate $P(Y = y)$ by counting the number of records that belong to class y, which will be referred as P_y, and the number N of records processed. Still, estimating $P(X = \vec{x}|Y = y)$ in not feasible. Using the NB assumption (i.e., that all attributes in X are independent given the class in Y) it is possible to estimate $P(X = \vec{x}|Y = y)$ by counting the number of records belonging to class y with x_i, which will be referred as $P_{x_i,y}$. Such assumption is also useful to deal with the dynamic feature space in data streams.

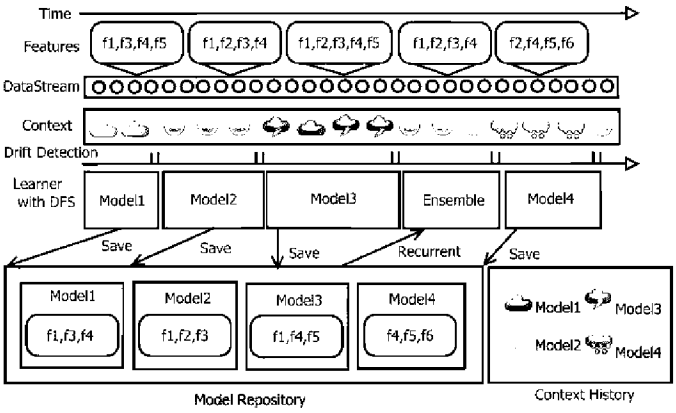

Fig. 1. MReC-DFS learning process.

B. Dynamic Feature Space

The feature space of the records $\vec{x} \in X_t$ observed in the data stream may be different over time. Also the relevance of each available feature may be different for each concept. We explore this idea to deal with recurring concepts in data streams. In this section, we describe how the features predictiveness is evaluated and some possible methods that can be used to select which features are worth saving in the model repository of previously learnt concepts.

1) Feature Evaluation and Selection: The base learner must represent the most relevant feature space for that time interval. It stores the estimation of the class probability $Pr(Y = y)$ and the estimation of each attribute given the class $Pr(X = \vec{x}|Y = y)$. These values are kept in a contingency table and are incrementally updated when new training records are available. Such information is enough to evaluate the predictive score of each individual feature, using popular methods as, the information gain, χ^2 or mutual information [15], [28].

After a change is detected if the situation immediately before corresponds to learning a new concept, the model that was being learnt is saved, but only its most predictive features are kept (i.e., the corresponding estimators for the attribute given the class). The features are evaluated and selected, however, many selection methods can be used for this task and according to the application some methods may be superior to other. Some possibilities are:

1) fixed N: select the top N highly scored features for each model. This strategy is simple and deterministic as it allows knowing in advance the amount of memory each model will consume. However, it is difficult to determine an adequate N *a priori*, particularly if the number of relevant features changes over time, as shown in Fig. 2(a). This method can easily ignore relevant features (e.g., $f2$ in t_i) while in other situations select irrelevant features. For example in t_{i+1} if we selected the top 4 features $f1$ (that is irrelevant) would be selected;

2) fixed Threshold, which defines the cut point between predictive and irrelevant features. This is more flexible than the previous method as it avoids storing irrelevant features that have low scores but end up being stored because they belong to the top N features. The major drawback of this measure is that it requires in advance the definition of an adequate threshold value, which can also change according to concepts over time. This is shown in Fig. 2(b) where for t_i the threshold value seems to seem adequate while for t_{i+1} it is too high and relevant attributes are lost;
3) adaptive threshold: we propose the usage of the percentile value to avoid the explicit definition of a threshold. The definition of the percentile is more natural than defining a fixed number of features and if the total number of features changes this method continues to capture the percentile parameter. The percentile value of the evaluation scores is dynamically calculated for the features of each model that is evaluated. The vector of scores is sorted and the estimated percentile position calculated as position $= p \times n + 1/100$ with $(0 < p < 100)$ and n as the number of features (not static). The position is rounded to the nearest integer and the percentile p value is obtained. This method's major advantage over the previous ones is that it can adaptively set the threshold according to the scores of a particular model. For example the 90th percentile for each model will vary according to the feature predictive scores, which make the method more robust to changes of scores across different models. This is shown in Fig. 2(c) where we can observe that the threshold changes in accordance with the evaluation score values.

Note that in situations where a threshold is used (whether fixed or adaptive) a maximum N sets an upper bound on the number of features that can be possibly saved and consequently the memory consumption of the approach.

2) Heterogeneous Feature Space Integration: The incremental nature of the feature evaluation-selection and learning process results that different features are considered predictive for each concept representation. In practice, this means that the learnt models may have a distinct feature space ($X_t \subseteq X$) between themselves, and that the learning system must be able to classify a new record considering different features over time. The base learner should be what has been referred in the literature as feature-based algorithms [15], which means that each feature contributes independently to calculate the resulting class. Examples of inherently feature-based algorithms are NB and the K-nearest neighbors. Considering feature interactions and other predictiveness measures is an open research problem that we plan to explore in future work.

MReC-DFS uses an ensemble of classifiers to deal with recurring concepts. These classifiers are likely to be represented by different feature spaces, as only the most relevant features are kept. When the models and by extent the ensemble are used to predict the class of a new record, this should be done in a homogenous feature space [15], [19], [28].

Some possible approaches to convert the feature space used during classification into a homogeneous feature space, exploiting the feature-based classifiers are:

1) fixed conversion: in this approach the feature space is fixed during the whole learning process. The feature evaluation and selection is executed with the first training records and will define the feature set for the learning process. This process losses much of the information from the data stream learning process where new features may emerge;
2) local conversion: in this approach each model uses the intersection of its feature space with the unlabeled record feature space, some features of the unlabeled records can be lost in this conversion. This has been the strategy used in [28];
3) homogenizing conversion: in this approach proposed in [19], each unlabeled record is classified using the union of the record and model features. Here no information is lost in the conversion process. Features that are not part of the unlabeled record will take a default value. In [19] it is shown that for novel classes and using their algorithm this method avoids misclassification into an existing class instance when compared with the local conversion method. However, in our approach the end results will be the same as the local conversion strategy, because only the intersection of features can actually be used in the classification of unlabeled records.

C. Context Representation and Similarity

The context representation and similarity used in MReC-DFS is inspired on the context spaces model [21], where a context state is represented as an object in a multidimensional Euclidean space. A context state c_i is defined as a tuple of N context attribute-values, $c_i = (a_1^i, ..., a_n^i)$ where a_n^i represents the value of context attribute a_n for the i^{th} context state c_i. For the purposes of this paper the degree of similarity between context states c_i and c_j, is based on the Euclidean distance.

The available context information depends on the learning environment and data mining problem. Context information can represent simple sensors (e.g., temperature and humidity) or a more complex context (e.g., season, location, and gait) defined by domain experts or inferred by other means beyond the scope of the problem discussed in this paper. In real-world scenarios as the records are usually timestamped, derived temporal features (e.g., period of the day, day of the week, month, and quarter) are normally used as contextual features.

D. Concept History

MReC-DFS saves past concept representations, thus it is important to have a memory efficient representation of concepts. NB as a base learner achieves this, as for each model m it only requires to store the estimation of the class P_y and the estimation of each attribute given the class $P_{x_i,y}$, we will refer to these as conceptual vectors, $cv = \{P_y, P_{x_i,y}\}$

1) Model Storage: Learnt models are kept stored so they can be reused in situations of recurrence. In such situations, the repository is searched for adequate models, that is, finding the

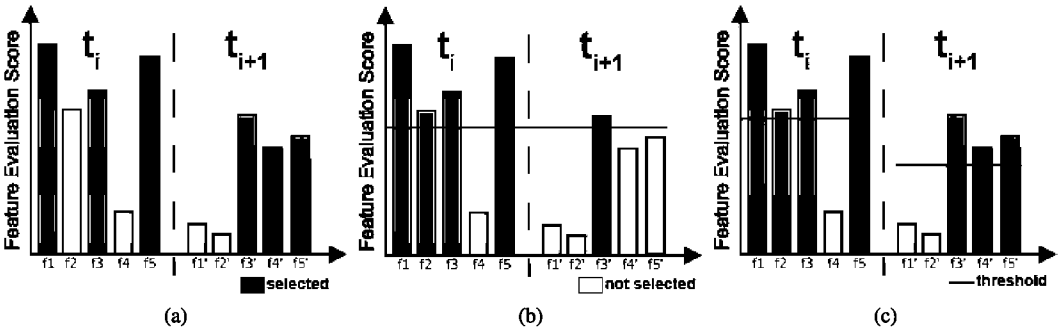

Fig. 2. Illustration of the feature selection methods over periods t_i and t_{i+1}.

models in the repository that represent the current underlying concept. If a good match is found, it is expected a reduction in the computational cost that comes associated with learning a new model and also improved adaptation to concept drift (i.e., better learning curve in terms of accuracy). For each classification model m in the model repository we store:

1) the concept representation: as the NB algorithm is used, this means to store the conceptual vectors of model m, that is represented as cv;
2) the conceptual vectors cv for m only contain information from the most relevant features and their evaluation scores as been described in IV-B as is referred as Selected($P_{x_i,y}$);
3) acc(m) = numCRecords$_m$/numRecords$_m$ is an estimate of the accuracy of m obtained during the period m was used, with numCRecords$_m$ as the number of correctly classified records by m and numRecords$_m$ the total number of records classified by m;
4) the timestamp t that records the period when a model m was used.

Most of the information that is kept, together with the context-concepts history is used in the selection of past models to represent the latest underlying concept. The memory consumption of the proposed approach is a function of the size used by the conceptual vectors cv, and as described previously, this depends on the number of features and classes that are considered, as more estimators need to be kept.

2) Context-Concepts Relation History: One of the main assumptions under our approach is that when a concept reappears normally the context previously associated with it also reappears. We take advantage of this fact to anticipate the adaptation to recurring concepts. Here, a description of how to create and represent the context-concepts relations history is presented.

Let m_j be the model learnt or used in a certain period j (i.e., that represents the underlying concept during that period j and $C_j = \{c_1, c_2, ..., c_n\}$ a sequence of n context records observed during this period j. The context-concepts history representation uses the NB algorithm to associate context with concepts. It is incrementally learnt from the sequence of context records C_j, where the model m_j identifier is used as the class label. This allows us to estimate the probability that a certain model m_k represents the current underlying concept given a certain context state c_i, we denote this estimation of probability as $h(m_k|c_i)$, similarly as has been previously explained for the base learner prediction of the class label given $\vec{x}$. As a consequence, we can keep an approximate and compact representation of the context-concepts relation history, without keeping the context records, which would be impossible due to the memory required. The maximum number of models (i.e., the number of classes in the context-concepts history) that we can store is predetermined according to the memory available. Note that when the ensemble is used the models that take part in the ensemble are associated with the current context.

3) Concept Similarity: To determine whether a certain model represents a new concept or a reappearing one a similarity measure is required. The Conceptual equivalence measure, that is based on the one proposed by Yang *et al.* [31] is used for this purpose. We should note due to the dynamic feature space it is likely that the models have a distinct feature space between themselves and also the records that are used to calculate the similarity score. For this reason the score is calculated using the intersection of the feature spaces of m_1 and m_2. Given two classification models m_1,m_2 and a sample dataset D_n of n records, it calculates for each instance X_i=($\vec{x}_i$, y_i) a score

$$\text{score}(X_i) = \begin{cases} +1 \ if \ \ m_1(\vec{x}_i) = m_2(\vec{x}_i) \\ -1 \ if \ \ m_1(\vec{x}_i) \neq m_2(\vec{x}_i) \end{cases} \quad (2)$$

that is used to represent the degree of equivalence between m_1 and m_2, that is an average continuous value score with range [−1,1], defined as

$$ce = \frac{\sum_{X_i \in D_n} \text{score}(X_i)}{N}. \quad (3)$$

The larger the output value, the higher the degree of conceptual equivalence. For the records in D_n it compares how m_1 and m_2 classify the records. The authors [31] argue that the accuracy and the conceptual equivalence degree are not necessarily positively correlated, as models can still achieve the same accuracy and misclassify different parts of the attribute space. We consider that if the obtained ce value is above a predefined threshold, the models are similar and thus represent the same underlying concept.

4) Ensemble Weighting: The main objective of the ensemble is to represent the current underlying concept and thus be able to classify the incoming records with high accuracy. One of the most important challenges of stream ensemble

approaches is on how to weight the models in the ensemble [22], [27]. We combine an accuracy weighted ensemble approach as proposed by Wang *et al.* [27] and the information learnt from the context-concept history described in IV-D.2 [2]. The model m_i accuracy is estimated using the mean square error (MSE). This measure estimates the error of the model for a window of records W_n. The MSE MSE_i for model m_i, using the window W_n of n records in the form of $(\vec{x}, y)$, where y is the true class label for that record, is defined as, $\text{MSE}_i = 1/|W_n| \sum_{(\vec{x},y)\in W_n}(1 - m_i^y(\vec{x}))^2$ where the error of m_i on record $(\vec{x}, y)$ is 1 - $m_i^y(\vec{x})$, and $m_i^y(\vec{x})$ is the probability given by m_i that $\vec{x}$ is an instance of class y.

Let MSE_r be the error estimation of a random classifier and is used as part of the filtering criterion (i.e., decide which classifiers are used in the ensemble). Models m_i with $\text{MSE}_i \times$ acceptanceFactor greater than MSE_r greater than a predefined threshold are not included in the ensemble. The acceptanceFactor is in the range [0,1] and was proposed in [22], to avoid that models that show minimal improvement from MSE_r participate in the ensemble.

Let acc_w and ctx_w be the weights assigned to the MSE_i calculated using dataset W_n and to the context-concepts history $h(m_i|c_o)$, representing the probability estimation that the current underlying concept is represented by m_i given the occurring context c_o (i.e., the most frequent context observed for the records in W_n)

$$\text{weight}(m_i) = \text{acc}_w \times (\text{MSE}_r - \text{MSE}_i) + ctx_w \times h(m_i|c_o). \quad (4)$$

The weights are normalized according with the number of models that take part in the ensemble. The sum of acc_w and ctx_w is 1.

E. Drift Detection and Adaptation

MReC-DFS requires to identify when drift occurs, and for this purpose it uses the method proposed by Gama *et al.* [7]. It assumes that periods of stable (i.e., the data distribution is stationary) concepts are observed followed by changes leading to a new period of stability with a different underlying concept. The error-rate (i.e., false predictions) of the learning algorithm is considered as a random variable from a sequence of Bernoulli trials. The binomial distribution gives the general form of the probability of observing an error. A warning and a drift level are defined, which represent the confidence levels (95% and 99% confidence, respectively) in a change of concept.

It should be noted that other methods for change detection can be used instead, without needing to change the proposed learning process/ adaptation strategy. Exploring other change detection methods for the particular problem of recurring concepts is an interesting line of research for future work.

One of the contributions of [1] is how to exploit contextual information when a concept change is detected. Here combined with the work in [2] using an ensemble approach to adapt to recurring concepts.

Algorithm 1 Data Stream Learning Process

Require: Data stream DS, ModelRepository MR

1: **repeat**
2: Get next record DS_i from DS;
3: prediction = *currentClassifier.classify*(DS_i);
4: *MR.updateStatistics(prediction)*;
5: *DriftDetection.update(prediction)*;
6: **switch** DriftDetection.level
7: **case** Normal
8: *history.train*(c_o,*currentClassifier)*;
9: **if** $\neg recurrent$ **then**
10: *currentClassifier.train*(DS_i);
11: *update.FS()*; //every x records
12: **end if**
13: **case** Warning
14: **if** $\neg MR.contains(currentClassifier)$ **then**
15: *MR.store(currentClassifier)*;
16: **end if**
17: **if** $history(c_o) > \rho$ **then**
18: *currentClassifier = MR.getPastModel*(c_o))
19: **end if**
20: *WarningWindow.add*(DS_i);
21: *newLearner.train*(DS_i);
22: **case** Drift
23: **repeat**
24: *WarningWindow.add*(DS_i);
25: *newLearner.train*(DS_i);
26: **until** $WarningWindow.size > \tau$ //Stability Period
27: **if** $\neg MR.containsEquivalent(newLearner)$ **then**
28: *currentClassifier = newLearner*;
29: **else**
30: *currentClassifier = MR.calcEnsemble*(c_o);
31: **end if**
32: **case** FalseAlarm
33: *WarningWindow.clear()*;
34: *newLearner.delete()*;
35: **end switch**
36: **until** END OF STREAM

F. Learning Process

The on-line learning process for the proposed learning system is detailed in Algorithm 1. The process proceeds as follows:

1) it continuously processes the records $DS_i = \{\vec{x}_i, y_i\}$ with $\vec{x} \in X_t$ as they appear in the Data Stream DS;
2) in line 3, currentClassifier represents the classifier that is currently being used to classify unlabeled records. In case of a recurring concept the ensemble is used. Its prediction (i.e., right or wrong) on X_i is passed to the drift detection method that identifies the current state of the learning process (i.e., stable, warning, or drift);
3) if the process is in the normal level (line 7), the record that represents the occurring context c_o is associated with the current model in the context-concepts relation history, and if the currentClassifier is new (i.e., not recurrent) it gets updated with the new training record.

The feature selection is executed periodically or simply when the model is stored;

4) in the case of warning level (line 13), if the repository does not have the currentClassifier, it is stored (please note that feature selection is also performed when storing, but here the irrelevant estimators are discarded). In addition (line 17), if the context-concepts relation history suggests a certain model with high probability, this model is reused and becomes the currentClassifier. This is a way to anticipate the adaptation that occurs in the drift level, but without requiring the collection of training records, it simply tries to predict what model would best represent the underlying concept given the current context. Still in this level (in line 20 and 21), a newLearner is updated with the training record and it is added to a warningWindow. This window contains the latest records (that should belong to the most recent concept), and will be used to calculate the conceptual equivalence and estimate the accuracy of stored models with the current concept;
5) when drift is signalled (line 22), until there are enough records (i.e., stability period) in the warningWindow the newLearner is updated. When the stability period is over (line 26) it is compared with repository models in terms of conceptual equivalence. If the current underlying concept is recurrent the ensemble is created reusing stored models that represent the recurring underlying concept, otherwise the newLearner is used;
6) a false alarm (line 32) is when a warning is signaled and then returns back to normal without reaching drift, in this case the warningWindow and the newLearner are cleared.

V. Evaluation

Experiments to test the feasibility and efficiency of the MReC-DFS learning system in terms of accuracy and resource consumption were performed. The implementation of the proposed learning system was developed in Java, using the massive online analysis (MOA) [3] environment as a test-bed. The MOA evaluation features (i.e., prequential-error [3]), the NB class as base learner (extended to deal with dynamic feature spaces) and the SingleClassifierDrift class were used. These provided a starting point to implement the specific components of our approach. The SingleClassifierDrift class implements the drift detection method (DDM) of [7] and adapts to it by learning a new classifier (i.e., discards previous concept representations). The features are evaluated using the information gain ratio method available in Weka [30], we also tested the χ^2 statistic and the information gain. The information gain ratio gives better results in the selection between predictive/irrelevant attributes and was used in the experiments.

The approach was tested using different high dimensional real-world text streams as data sources. The real-world datasets used are well known for text classification [16]. Since we are interested in scenarios of recurring concepts, we created an experimental environment where the user interest changes according to context and where the learning task is to correctly predict the current user interest in a certain document.

A. Evaluation Metrics

In traditional batch learning algorithms with limited training sets, techniques, such as cross-validation, leave-one-out or bootstrap are the standard methods of evaluation. Cross-validation is appropriate for limited size datasets, generated by stationary distributions, and assuming that examples are independent. In data streams scenarios, where data is potentially infinite, the distribution generating examples and the classification models evolve over time, cross-validation and other sampling strategies are not applicable.

When evaluating in the data stream scenario, the major goal is to establish a measure of the accuracy over time. One possible solution involves taking snapshots at different times during the induction of the model to see how much the model improves as additional records are processed. The evaluation procedure of a learning algorithm must determine which records are used for training the algorithm, and which are used to test the classification model created by the algorithm. To evaluate a classification model in a data stream scenario, two possible alternative procedures presented in the literature are:

1) holdout an independent test set: In traditional batch learning, if the data reaches a scale where cross-validation is too time consuming, it is often accepted to measure performance on a single holdout set instead. In a data stream scenario, the current classification model is applied to the test set, at regular time intervals (or set of records). The loss estimated in the holdout is an unbiased estimator. This is most useful when the division between train and test sets has been predefined, so that results from different studies can be directly compared (e.g., in popular data mining competitions);
2) interleaved Test-Then-Train or Prequential (Predictive Sequential): In the Prequential approach, the error of a model is calculated from a sequence of records. Each individual record can be used to test the model before it is used for training, the prediction is based only on the record attribute-values (i.e., ignores the class value). Using this order, test then train, the model is tested on new records (i.e., records it has not seen before). The main advantage with this approach is that no holdout set is needed for testing, making use of all the available training data. In addition, in situations where the underlying concept is stable, it ensures a smooth plot of accuracy over time, as each individual record will become increasingly less significant to the overall average [8].

The advantages of the prequential test-then-train evaluation procedure for the data stream scenario motivate its usage in our experiments.

B. Datasets

In the following, we describe the datasets used in our experimental study.

1) Email List: The email list (elist) dataset[1] [16] is a stream of email messages from topics that are labeled by a user as interesting or junk according to his preferences. The dataset is composed of 1.500 records and 913 attributes which represent words that occur in the corpus with frequency higher than ten. To simulate concept drift and recurring concepts the dataset contains two recurrent concepts that change every 300 records. The first concept represents messages where user is only interested in medicine and in the second concept the interest changes to space and baseball.

2) Reuters: The Reuters dataset[2] is usually used to test text categorization approaches. It contains 21.578 news documents from the Reuters news agency collected from its newswire in 1987. From the original dataset two different datasets are usually used, the R52 and R8. R52 is the dataset with the 52 more frequent categories, whereas R8 only uses the 8 more frequent categories. The R8 dataset has 5.485 training documents and 2.189 testing documents. In our experiments from R8 we use the two most frequent categories earn (2.229 documents), acq (3.923 documents) and others (a group with the 6 remaining categories, with 1.459 documents).

3) WebKD: The webKD dataset[1]contains web pages of computer science departments of various universities. The corpus contains 4.199 pages, 2.803 training pages, and 1.396 testing pages, which are categorized into: project, course, faculty, and student.

C. Recurrent Concepts

Similarly to what has been proposed in [16] with the elist dataset using the 20 Newsgroups data to deal with recurrent concepts, we created a MOA data stream generator that wraps the last three aforementioned text documents. The generator simulates document filtering according to user interest. Each record generated from the stream represents binary labeled (i.e., interesting or irrelevant) document from the collection. The labels are calculated dynamically when matching the original document classification with the current user preferences. For example, in webKD data there are four classes, to simulate a user that is only interested in pages about course and project, all the documents that belong to these topics will be labeled as interesting. In addition, the generator allows the definition of the current underlying concept by changing the topics of interests. In the scope of our experiments, this allows control over concept changes and user preferences may be repeated to simulate recurring concepts.

1) Context-Concept Relations: For simplicity and without loss of generality the context is generated as a stream of context records within a context space C that uses two context attributes $C = \{\text{location}, \text{weekDay}\}$. With location $\in$ $\{\text{work}, \text{home}\}$, weekDay $\in \{\text{week}, \text{weekend}\}$.

In our experiments, the target concepts of the learning process represent user interests in a set of topics. These interests are associated with context. For example, a user at home is not interested in science, but documents related to sport and cinema.

[1]http://mlkd.csd.auth.gr/concept_drift.html

[2]http://www.cs.umb.edu/ smimarog/textmining/datasets/index.html

Table I summarize for the aforementioned datasets, what are the user interests and which is the associated context for each 500 records period.

In the elist dataset the context (work, week) occurs with the first concept (medicine), whereas (home, weekend) occurs with the second concept (space+baseball).

D. Experiments

MReC-DFS is compared with the different learning algorithms:

1) incremental NB, a single NB processes the data stream records. The method can inherently deal with the dynamic feature space and is used as a baseline and control in our experiments. However, since no selection is performed, if the combined feature space has high dimensionality too many estimators must be kept, which can limit the method performance and its memory consumption;
2) moving Window (MW) NB, a single NB processes the data stream records, the same as the previous method but only considers the records within a time window of fixed size;
3) single classifier drift, MOA implementation of the DDM in [7]. We used the incremental NB as base classifier, because it can learn in a dynamic feature space, however the same limitations in performance that apply to the NB may be applied here if the dimensionality is high, because no feature selection is used;
4) DWM [17], MOA implementation of DWM. Again the NB as base classifier was used. The default parameters in MOA for this algorithm are beta set to 0.5, the period between expert removal, creation, and weight update p, set to 50 records and gamma that is the minimum fraction of weight per model set to 0.01;
5) ozaBoostAdwin [4] MOA implementation of the Online boosting algorithm with ADWIN. Again the NB as base classifier was used. The default parameters in MOA for this algorithm are ten for ensemble size and 0.002 for the Delta of Adwin change detection.

MReC-DFS is instantiated in the following versions.

1) rec, serves as baseline method, it tracks recurring concepts using context information and the base learner is the NB algorithm, no feature selection is performed;
2) rec+DFS-TopN, same as Rec but feature selection is performed keeping the top N features (according to their evaluation score) for a given model;
3) rec+DFS-Threshold, same as Rec but feature selection is performed keeping the features with score above a predefined threshold;
4) rec+DFS-Adaptive, same as Rec but the feature selection is performed using an adaptive threshold as described in Section IV-B.1, for the experiments we used the 75th percentile.

   As parameters, the MSE and context weights in the ensemble creation were set to 0.5, the stability period used was 30 records, the context training period was 900 records and the context history threshold was set to 0.3,

TABLE I
INTERESTS ASSOCIATED WITH CONTEXT OVER TIME

Context/ Dataset	work week	home week	work week	home week	home weekend	work week	home week	work weekend	work week	home week
R8	earn	acq	earn	acq	others	earn	acq	others	earn	acq
WebKD	project course	student	project course	student	faculty	project course	student	faculty	project course	student
Start	0	500	1000	1500	2000	2500	3000	3500	4000	4500

TABLE II
MEMORY SAVED USING FEATURE SELECTION

		Memory consumed		
Dataset	Method	Memory	savedMemory	Accuracy
elist	Rec	501.960	0%	0.736
	Rec+DFS(0.01)	368.272	26.6%	0.717
	Rec+DFS(300)	332.072	33.8%	0.697
	Rec+DFS(Adap)	220.024	**56.2%**	**0.761**
R8	Rec	681.808	0%	0.796
	Rec+DFS(0.05)	485.584	**28.7%**	0.807
	Rec+DFS(300)	489.456	28.2%	0.791
	Rec+DFS(Adap)	489.912	28.1%	**0.829**
webKD	Rec	892.184	0%	0.780
	Rec+DFS(0.05)	346.240	**61.1%**	0.770
	Rec+DFS(300)	444.216	50.2%	0.772
	Rec+DFS(Adap)	443.088	50.3%	**0.797**

whereas the similarity threshold was 0.6. These parameters were not fine tuned for each dataset. The parameters of each particular method/variation are shown in tables and figures with the results.

E. Results and Discussion

To assess MReC-DFS efficacy and efficiency, in relation to the different methods, we measured the Prequential (as defined in V-A) predictive accuracy, precision, recall, number of instances processed to train the models and total memory consumed (only between the methods that store models).

In our experiments we did not measure the running time between methods as all the methods have linear temporal complexity and can process the data as it arrives in the data stream. Therefore, their difference in running time would not be meaningful under the scope of this paper. However, we consider the memory consumption to evaluate the tradeoff between accuracy and memory used.

MOA allows to easily estimate the Prequential-error of the learning process and the figures show the average accuracy over time considering a sliding window of 50 records. To assess the statistical significance of the results we use a paired t-test between the accuracy estimation at every window the last record in the previous window is discarded (i.e., every 50 records). The results were tested at a 0.95 significance level. In the figures, the vertical lines indicate a change in concept. In addition, the figures scale changes to increase the readability of the curves, so for better comparison the (Rec) approach is shown in both figures (left and right).

1) Elist: Table III shows the experimental results for the elist dataset, we observe that the methods that explicitly explore concept Rec outperform the other methods. Between the methods (Rec) and variations, we see that due to exploiting recurring concepts all these require to process less records and achieve good accuracy at the cost of increased memory consumption (table II). Consequently, there is a good trade

TABLE III
RESULTS OF ALL METHODS USING THE ELIST DATASET

Method	Accuracy	Precision	Recall	ProcInst
NB	0.542	0.507	0.692	1500
MW(100)	0.661	0.632	0.652	1500
OzBoostAdwin	0.576	0.560	0.425	1500
DWM	0.673	0.632	0.717	1500
DDM	0.707	0.664	0.754	1500
Rec	0.736	0.693	0.778	**630**
Rec+DFS(0.01)	0.717	0.674	0.761	633
Rec+DFS(300)	0.697	0.654	0.742	908
Rec+DFS(Adap)	**0.761**	**0.717**	**0.807**	1020

off in the accuracy and the memory consumed. However, as it can be seen when comparing the base method with the ones that use feature selection, the later achieve reduced memory consumption. Considering the three feature selection methods, we see that selecting the top N (Rec-DFS(TopN)) for these concepts results in the worse accuracy value. This is a consequence of the concepts and N, as some useful features may be lost and the models (with selected features) fail to represent accurately the recurring concepts in comparison with other models where a more flexible feature selection method is used. For example, the Rec-DFS(Threshold) method results in the second best accuracy within the feature selection methods, however determining the threshold is not trivial, particularly in the context of online learning, as the threshold value that is more adequate can change with concepts. The adaptive threshold method proposed in this paper, tries to address such issue, we observe that it achieves the best results for this dataset. In addition, these results are statistical significance in relation to the (DDM) method with a p-value of 9.225e-07 and (Rec) with a p-value of 0.01827.

In Fig. 3, we can see on the left the methods without feature selection, our proposal that exploits recurring concepts is shown in both figures and in the right figure the different feature selection methods are shown. The left figure shows that the MW (where its model always refers to the most recent records in the window, which makes it able to adapt to the underlying concept) achieves the worse performance followed by the DWM. The DWM achieves very good accuracy for the first two concepts. However, no explicit drift detection is used, for this reason the DDM and Rec achieve better overall results as they can explore more training records. While learning the two concepts both methods achieve the same results (until record 600), but after learning the underlying concepts, the method that exploits Rec is able to reuse these models and adapt more efficiently to the concept changes. In the figure on the right we can observe that all the methods have similar behavior but that Rec and Rec-DFS(Adaptive) achieve the best curve and adapt better. Rec because it exploits Rec and keeps all the relevant features, however at the cost of

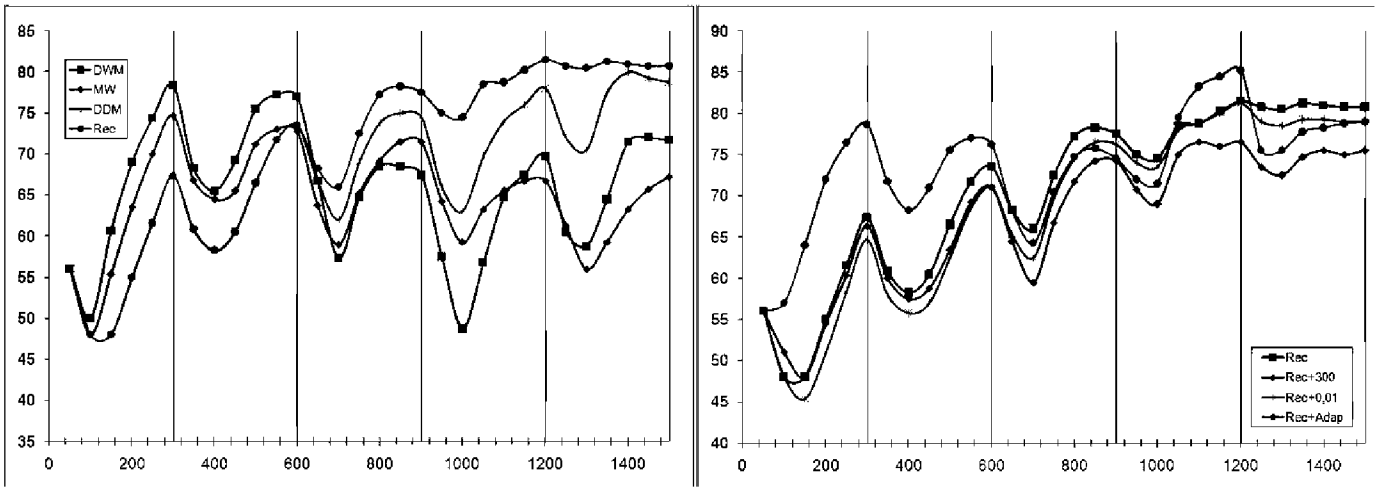

Fig. 3. Accuracy of the methods using the elist dataset.

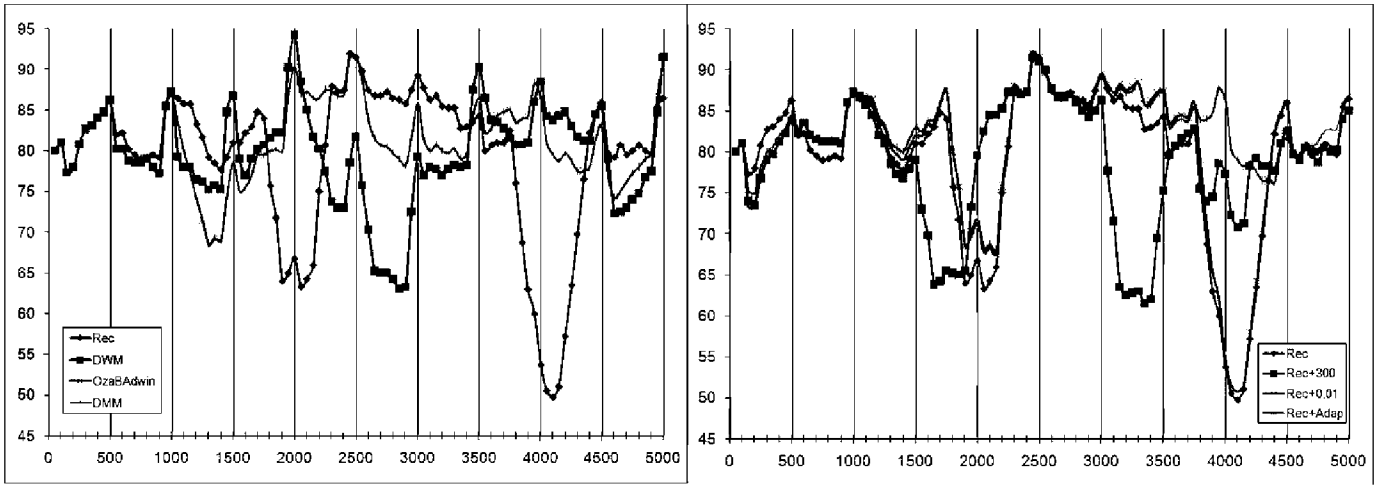

Fig. 4. Accuracy of the methods using the R8 dataset.

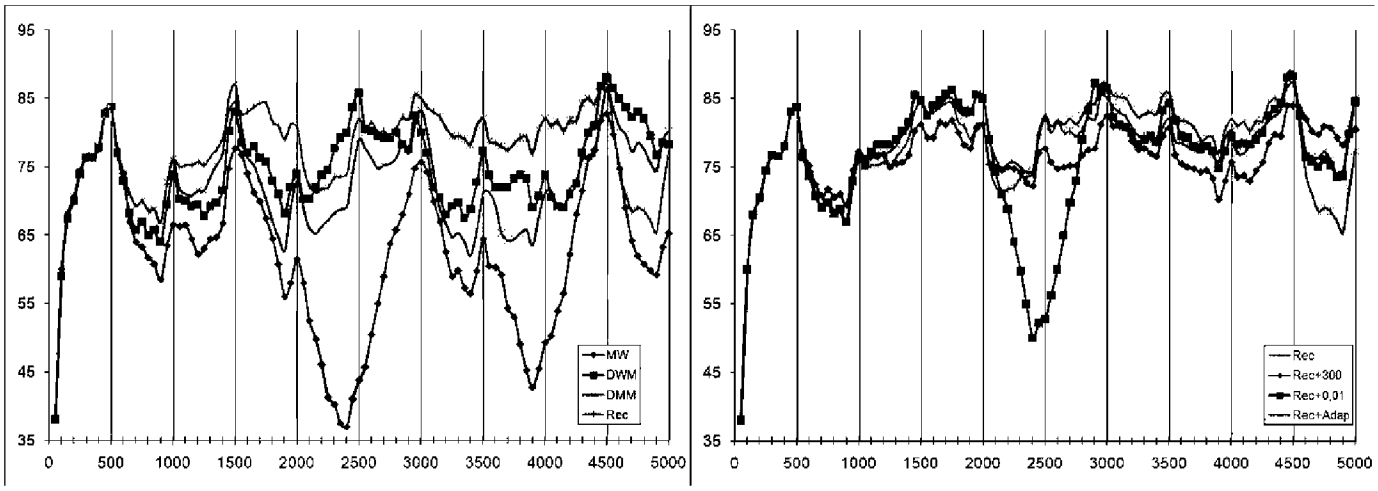

Fig. 5. Accuracy of the methods using the webKD dataset.

more memory consumed, and the Rec-DFS(Adaptive) because for each model the feature selection threshold is calculated according to the evaluation values and not by some fixed value determined *a priori.*

2) Reuters R8: Table IV shows the experimental results for the Reuters R8 dataset. We see that again the feature selection helps to improve the accuracy of the results. Because of the existence of three different concepts in this dataset, we can

TABLE IV
RESULTS OF ALL METHODS USING THE R8 DATASET

Method	Accuracy	Precision	Recall	ProcInst
NB	0.517	0.404	0.651	5000
MW(250)	0.657	0.529	0.684	5000
OzBoostAdwin	0.718	0.624	0.596	5000
DWM	0.800	0.712	0.772	5000
DDM	0.816	0.705	0.864	5000
Rec	0.796	0.689	0.818	1504
Rec+DFS(0.05)	0.807	0.702	0.829	**1503**
Rec+DFS(300)	0.791	0.681	0.816	2192
Rec+DFS(Adap)	**0.829**	**0.731**	**0.850**	2247

TABLE V
RESULTS OF ALL METHODS USING THE WEBKD DATASET

Method	Accuracy	Precision	Recall	ProcInst
NB	0.569	0.380	0.372	5000
MW(250)	0.630	0.472	0.526	5000
OzBoostAdwin	0.617	0.434	0.338	5000
DWM	0.749	0.625	0.707	5000
DDM	0.732	0.592	0.740	5000
Rec	0.780	0.664	0.746	1524
Rec+DFS(0.05)	0.770	0.660	0.704	**1517**
Rec+DFS(300)	0.772	0.655	0.732	2050
Rec+DFS(Adap)	**0.797**	**0.689**	**0.760**	2023

observe in Table II that the memory consumption increases in the Rec approach, whereas this is minimized when we look at the feature selection methods and a very good tradeoff between accuracy and memory consumed is achieved by the adaptive threshold method. This is also seen in Fig. 4 where Rec-DFS(Adaptive) adapts much better to the recurring concept changes. We observe that for the 3rd concept (which is difficult to track, because of the accuracy dip around 4.000), Rec-DFS(Adaptive) obtains the best results as it adapts smoothly to the concept change, which is even better than the Rec method, which can be explained by the number of records processed that maybe is not enough to represent efficiently all the underlying concepts, particularly the aforementioned 3rd concept. Note that, once more, the fixed feature selection Rec-DFS(TopN) method is the worse of the sophisticated methods as it sometimes fails to adapt to recurrence, maybe due to loss of important features. Concerning the DDM, it clearly achieves good performance with low memory consumption; however, its main drawback is that the base learner always relearns all concepts from scratch and needs to process all the records in the stream. In addition, the ensemble methods DWM and OzBoostAdwin also achieve good accuracy for this dataset as it can be seen in Table II and Fig. 4. Finally, when comparing the statistical significance of Rec-DFS(Adaptive) results against the best competitors, we obtained for (DDM) method a p-value of 0.03446 and for (Rec) a p-value of 2.328e-05.

3) WebKD: Table V shows the experimental results obtained for the webKD dataset. Again a clear advantage is seen for the methods that exploit Rec, these obtain better predictive accuracy and process fewer records than the other methods. Table II shows that the memory cost associated with saving models is minimized by the feature selection methods. These show a minimal loss in accuracy which is the result of an effective feature selection. The differences between the feature selection methods can be attributed to the relative success of these in keeping the most relevant features. The adaptive threshold selection method gives the best accuracy while reducing the memory cost to half. Fig. 5 shows that the curves of the (Rec) methods are superior to the other methods; however, the feature selection methods are not free from drawbacks, as is seen for the fixed threshold method around record 2.000, this shows that the method failed to adapt to Rec maybe due to loss of relevant features which has an impact on measuring the conceptual equivalence. However, we should reinforce that no fine tuning was performed between the experiments to alleviate such drawbacks. Still, within the Rec methods the Rec-DFS(Threshold) was the one with lowest memory consumption and less processed records. This shows again the importance of the tradeoff between accuracy and resources in data stream learning systems. Finally, when comparing the statistical significance of Rec-DFS(Adaptive) results against (DWM) method we obtained a p-value of 1.479e-12 and a p-value of 1.874e-08 for (Rec).

VI. CONCLUSION

This paper proposes MReC-DFS, a data stream learning system in a dynamic feature space. MReC-DFS addresses concept changes by monitoring the error rate of the learning process. Moreover, to deal with recurring changes in concept, instead of relearning from scratch previously learnt concepts, models from past concepts are saved so when a similar concept reappears they can be reused. Context is exploited to improve the system adaptation to recurring concepts in situations where context is associated with the target concepts. To minimize the memory cost that comes associated with the benefits of the approach, and based on the definition of predictive/irrelevant attributes, only the most predictive features are kept. We discuss possible methods for selecting the most relevant features and propose an adaptive threshold feature selection solution.

To evaluate MReC-DFS and the possible feature selection methods, an implementation of the proposed system has been developed. We run tests using a known dataset for recurring concepts and with a generator of recurring concepts that uses two popular datasets from text mining. We tested and compared nine different methods in our experiments. The experimental results show that feature selection can be used to minimize the cost associated of learning recurring concepts in data streams with a dynamic feature space. Moreover, the results show that the adaptive threshold method proposed in this paper outperforms the other methods and achieves the best tradeoff between accuracy and resources (i.e., the number of processed instances by the base learner and the total memory consumed).

In the future work, it would be interesting to study how this proposal can be used to learn from multiple data streams.

REFERENCES

[1] J. Bartolo Gomes, E. Menasalvas, and P. Sousa, "Tracking recurrent concepts using context," in *Proc. 7th Int. Conf. RSCTC*, 2010, pp. 168–177.

[2] J. Bartolo Gomes, E. Menasalvas, and P. Sousa, "Learning recurring concepts from data streams with a context-aware ensemble," in *Proc. ACM SAC*, 2011, pp. 994–999.

[3] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, May 2010.

[4] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 139–148.

[5] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[6] J. Gama and P. Kosina, "Tracking recurring concepts with meta-learners," in *Proc. 14th Portuguese Conf. Artif. Intell.*, Oct. 2009, p. 423.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence* (Lecture Notes in Computer Science). New York, NY, USA: Springer-Verlag, 2004, pp. 286–295.

[8] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, 2013.

[9] J. B. Gomes, P. A. C. Sousa, and E. Menasalvas, "Tracking recurrent concepts using context," *Intell. Data Anal.*, vol. 16, no. 5, pp. 803–825, 2012.

[10] P. M. Gonçalves and R. S. M. Barros, "RCD: A recurring concept drift framework," *Pattern Recognit. Lett.*, vol. 34, no. 9, pp. 1018–1025, 2013.

[11] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Mach. Learn.*, vol. 32, no. 2, pp. 101–126, 1998.

[12] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011.

[13] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "New management operations on classifiers pool to track recurring concepts," in *Data Warehousing and Knowledge Discovery*. New York, NY, USA: Springer-Verlag, 2012, pp. 327–339.

[14] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 97–106.

[15] I. Katakis, G. Tsoumakas, and I. Vlahavas, "On the utility of incremental feature selection for the classification of textual data streams," in *Advances in Informatics*. New York, NY, USA: Springer-Verlag, 2005, pp. 338–348.

[16] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: An application to email filtering," *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, 2010.

[17] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.

[18] P. Li, X. Wu, and X. Hu, "Mining recurring concept drifts with limited labeled streaming data," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, p. 29, 2012.

[19] M. Masud, Q. Chen, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Classification and novel class detection of data streams in a dynamic feature space," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2010, pp. 337–352.

[20] M.D. Muhlbaier, A. Topalis, and R. Polikar, "Learn^{++}.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[21] A. Padovitz, S.W. Loke, and A. Zaslavsky, "Towards a theory of context spaces," in *Proc. 2nd IEEE Annu. Conf. Pervas. Comput. Commun. Workshops*, Mar. 2004, pp. 38–42.

[22] S. Ramamurthy and R. Bhatnagar, "Tracking recurrent concept drift in streaming data using ensemble classifiers," in *Proc. 6th Int. Conf. Mach. Learn. Appl.*, 2007, pp. 404–409.

[23] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, 2012.

[24] W. N. Street and Y. S. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 377–382.

[25] A. Tsymbal, "The problem of concept drift: Definitions and related work," Dept. Comput. Sci., Trinity College, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.

[26] P. D. Turney, "Exploiting context when learning to classify," in *Proc. ECML*, 1993, pp. 402–407.

[27] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 226–235.

[28] B. Wenerstrom and C. Giraud-Carrier, "Temporal data mining in dynamic feature spaces," in *Proc. 6th Int. Conf. Data Mining*, 2007, pp. 1141–1145.

[29] G. Widmer, "Tracking context changes through meta-learning," *Mach. Learn.*, vol. 27, no. 3, pp. 259–286, 1997.

[30] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2005.

[31] Y. Yang, X. Wu, and X. Zhu, "Mining in anticipation for concept change: Proactive-reactive prediction in data streams," *Data Mining Knowl. Discovery*, vol. 13, no. 3, pp. 261–289, 2006.

[32] I. Zliobaite, "Learning under concept drift: An overview," Faculty Math. Inf., Vilnius Univ., Vilnius, Lithuania, arXiv: 1010.4784, 2010.